Compulsory exercise 1: Group 28

TMA4250 Spatial Statistics V2022

Nora Røhnebæk Aasen, Elias Klakken Angelsen

18 februar, 2022

```
set.seed(2824)
```

Problem 1: GRFs - model characteristics

a)

The correlation function ρ is supposed to be a positive semi-definite function. This means that we require

$$\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j \rho(s_i, s_j) \geq 0, \qquad \forall a_1, \ldots, a_m \in \mathbb{R},\ \forall s_1, \ldots, s_m \in \mathcal{D}$$

This is a requirement in order to ensure well-defined (i.e. positive) variance, since

$$Var\left(\sum_{i=1}^{m} b_i X(s_i)\right) = \sum_{i=1}^{m}\sum_{j=1}^{m} b_i b_j C(s_i, s_j) = \sum_{i=1}^{m}\sum_{j=1}^{m} b_i b_j \sigma^2 \rho(s_i, s_j) = \sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j \rho(s_i, s_j),$$

where C is the covariance function, $\sigma^2 = C(0)$ and $a_i = b_i\sigma$ and $a_j = b_j\sigma$.

Below we plot the correlation function $\rho(h) = C(h)/C(0)$ and the semi-variogram function $\gamma(h) = \frac{1}{2}[Var(X(h) - X(0))]$ for 4 different covariance functions, and for two different choices of σ^2.

The two covariance models are Matérn(note that this model differs from the one used in cov.spatial in R, since what they call ϕ is $\frac{a}{\sqrt{8\nu}}$ for us):

$$C(h; a, \nu) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{8\nu}h}{a}\right)^{\nu} K_\nu\left(\frac{\sqrt{8\nu}h}{a}\right)$$

and powered exponential:

$$C(h; a, \alpha) = \exp\left\{-\left(\frac{h}{a}\right)^{\alpha}\right\}, \quad 0 < \alpha \leq 2$$

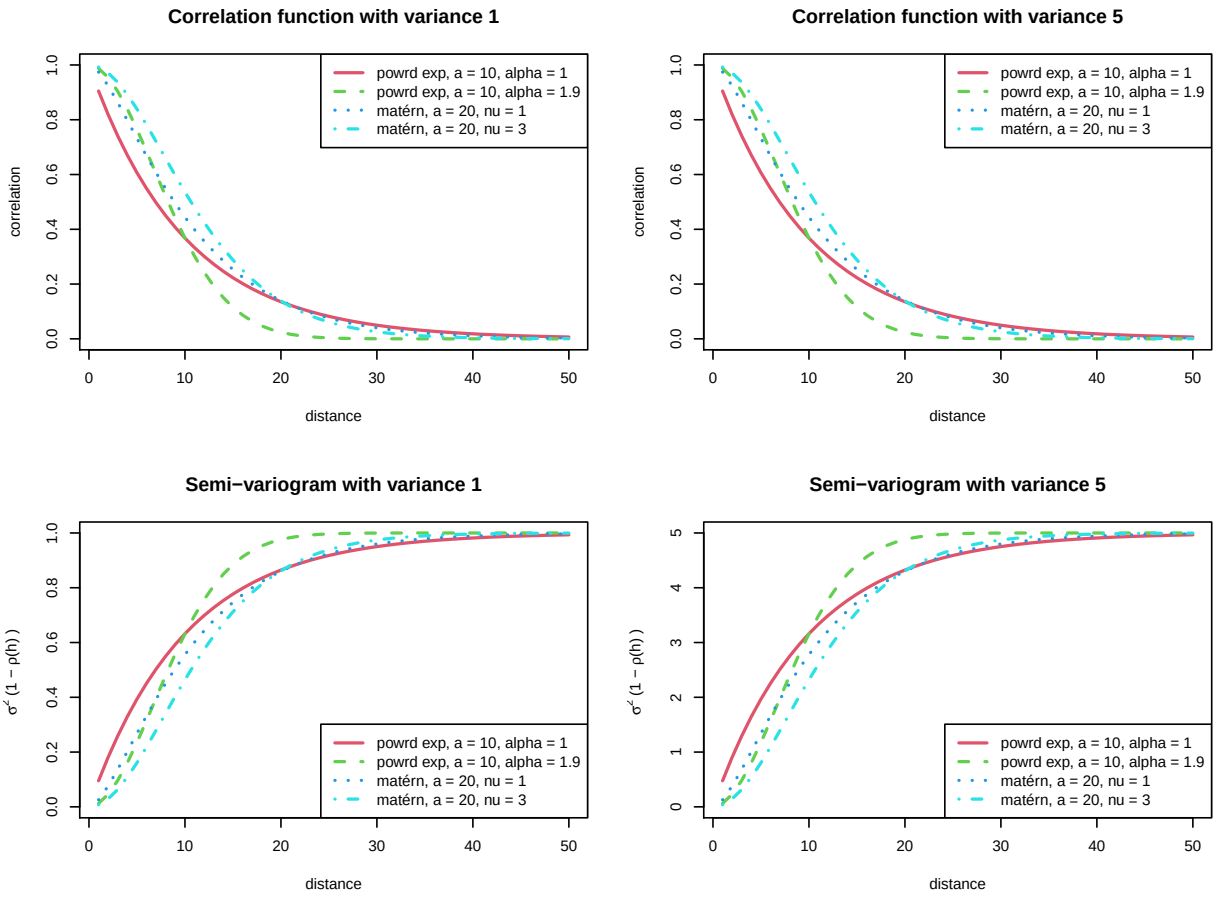
These can be plotted in one dimension since they are stationary in this case, and, additionally, isotropic since $\mathcal{D} \subset \mathbb{R}$.

We plot the correlation functions and the variograms for our models.

```r
# define a function that returns a vector with the values of the correlation function
corr.spat <- function(x,type,sigma,a,param){
  x = cov.spatial(x, cov.model=type, cov.pars=c(sigma,a), kappa = param)
  return(x/sigma)
}


# define a function that returns a vector with the values of the semi-variogram
semi.v <- function(x,type,sigma,a,param){
  return(sigma*(1-corr.spat(x,type,sigma,a,param)))
}


# define the discretized domain D and parameters
D_domain = seq(1,50,1)

sigma_sqr = c(1,5)
alpha_exp = c(1,1.9)
nu_matern = c(1,3)
ran_matern = 20/sqrt(8*nu_matern)

param = expand.grid(sigma_sqr,c(alpha_exp,nu_matern))




# Make empty matrices to fill with data
pwd_corr_mx = matrix(NA,nrow = 50,ncol = 4)
matern_corr_mx = matrix(NA,nrow = 50,ncol = 4)

pwd_sv_mx = matrix(NA,nrow = 50,ncol = 4)
matern_sv_mx = matrix(NA,nrow = 50,ncol = 4)




# Fill the matrices with the data from the models

for (i in (1:4)){
    # define the covariance matrix of powered exponential model
    pwd_corr_mx[,i] = corr.spat(D_domain, "powered.exponential",param[i,1],10,param[i,2])

    # define the covariance matrix of matérn model
    matern_corr_mx[,i] = corr.spat(D_domain, "matern",param[4+i,1],ran_matern[ceiling(0.5*i)],param[4+i

    # define the semi-variogram matrix of powered exponential model
    pwd_sv_mx[,i] = semi.v(D_domain, "powered.exponential",param[i,1],10,param[i,2])

    # define the semi-variogram matrix of matérn model
    matern_sv_mx[,i] = semi.v(D_domain, "matern",param[4+i,1],ran_matern[ceiling(0.5*i)],param[4+i,2])
```

```
}



# Change the names of the columns
colnames(pwd_corr_mx) = c("cov1_alp1", "cov5_alp1", "cov1_alp1.9", "cov5_alp1.9")
colnames(pwd_sv_mx) = c("cov1_alp1", "cov5_alp1", "cov1_alp1.9", "cov5_alp1.9")
colnames(matern_corr_mx) = c("cov1_nu1", "cov5_nu1", "cov1_nu3", "cov5_nu3")
colnames(matern_sv_mx) = c("cov1_nu1", "cov5_nu1", "cov1_nu3", "cov5_nu3")




# Store and plot the correlation functions and semi-variograms

corr_1 = cbind(pwd_corr_mx[,1],pwd_corr_mx[,3],matern_corr_mx[,1],matern_corr_mx[,3])
corr_5 = cbind(pwd_corr_mx[,2],pwd_corr_mx[,4],matern_corr_mx[,2],matern_corr_mx[,4])


#Plot the four plots together
par(mfrow = c(2,2), cex = 0.6)


#Plotting correlation functions for variance = 1
matplot(as.data.frame(corr_1), type = 'l', col = c(2:5), lwd = 2, ylim = c(0,1), main = paste('Correlati

legend('topright', c("powrd exp, a = 10, alpha = 1", "powrd exp, a = 10, alpha = 1.9", "matérn, a = 20,




#Plotting correlation functions for variance = 5
matplot(as.data.frame(corr_5), type = 'l', col = c(2:5), lwd = 2, ylim = c(0,1), main = paste('Correlati

legend('topright', c("powrd exp, a = 10, alpha = 1", "powrd exp, a = 10, alpha = 1.9", "matérn, a = 20,




# Store semi-variogram data for plotting
semi.v_1 = cbind(pwd_sv_mx[,1],pwd_sv_mx[,3],matern_sv_mx[,1],matern_sv_mx[,3])
semi.v_5 = cbind(pwd_sv_mx[,2],pwd_sv_mx[,4],matern_sv_mx[,2],matern_sv_mx[,4])



#Plotting semi-variograms for variance = 1

matplot(as.data.frame(semi.v_1), type = 'l', col = c(2:5), lwd = 2, ylim = c(0,1), main = paste('Semi-va

legend('bottomright', c("powrd exp, a = 10, alpha = 1", "powrd exp, a = 10, alpha = 1.9", "matérn, a = 1
```

```r
#Plotting semi-variograms for variance = 5

matplot(as.data.frame(semi.v_5), type = 'l', col = c(2:5), lwd = 2, ylim = c(0,5), main = paste('Semi-va

legend('bottomright', c("powrd exp, a = 10, alpha = 1", "powrd exp, a = 10, alpha = 1.9", "matérn, a = 1
```



The correlation plots are equal, as the varying factor σ^2 doesn't affect the correlation. The semi-variogram depends on the covariance function through $\gamma(h) = [C(0) - C(h)]$, and since we have $C(0) = \sigma^2$ and $\rho(h) = C(h)/C(0)$, we can show

$$\gamma(h) = [C(0) - C(h)] = [\sigma^2 - \sigma^2\rho(h)] = \sigma^2[1 - \rho(h)].$$

This relationship is visible through the plots. Furthermore the range of the covariance function and the sill can be read from the semi-variogram plots. The range is the distance at which dependence is negilable, which is 10 and 20 in our case respectively. We see that the Matérn covariances has a longer range than the powered exponential. The sill is the marginal variance, i.e. σ^2, and can be found around the point where the semi-variogram decays. In our case, this is 1 and 5 as expected.

The behavior around 0 indicates the differentiation of the covariance function and whether it is quadratic mean continuous or not. From the graph we would assume the powered exponential with range 10 and $\alpha = 1$ is not continuous, while the others are. We need $2k$ times differentiability in 0 for the covariance function to be k times quadratic mean differentiable. By looking we could guess that Matérn with range 20 and $\nu = 1$ is not differentiable, whereas the two last ones might be at least one time quadratic mean differentiable.

For the variogram function $2\gamma(h)$, the relationship to the covariance function is

$$2\gamma(h) = 2[C(0) - C(h)] = 2\sigma^2[1 - \rho(h)].$$

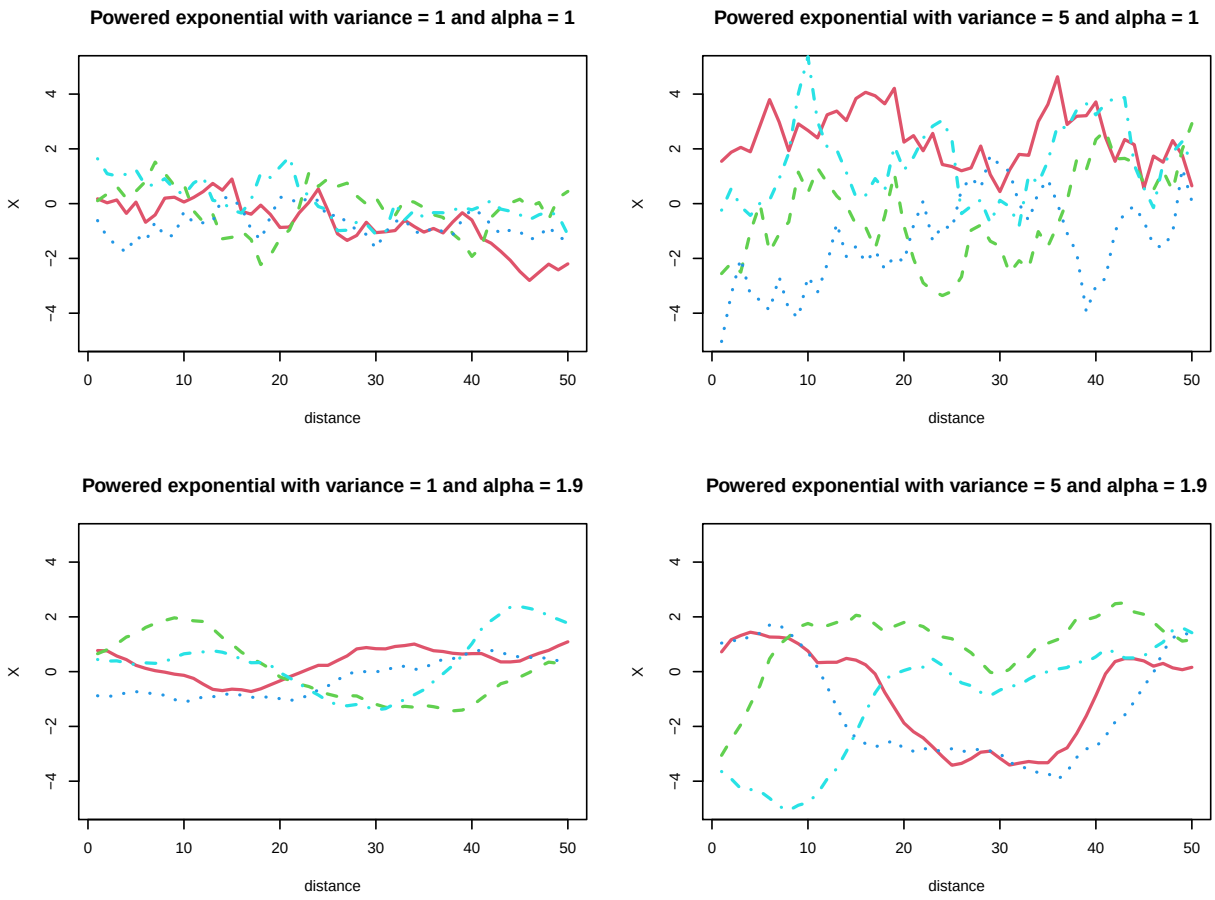
b)

Since X is a Gaussian random field, we know that its distribution is completely determined by its mean and covariance matrix. Furthermore, the vector $\mathbf{X} = (X(s_1), \ldots, X(s_{50}))$ is also normal and has a multivariate normal distribution. Therefore, $\mathbf{X}(s) \sim \mathcal{N}_{50}(\mu, \Sigma)$, where $\mu = E[\mathbf{X}(s)] = 0$ in our case, and $\Sigma_{ij} = Cov(X(s_i), X(s_j)) = \sigma^2\rho(\|s_i - s_j\|)$ for $s \in \mathcal{D}$.

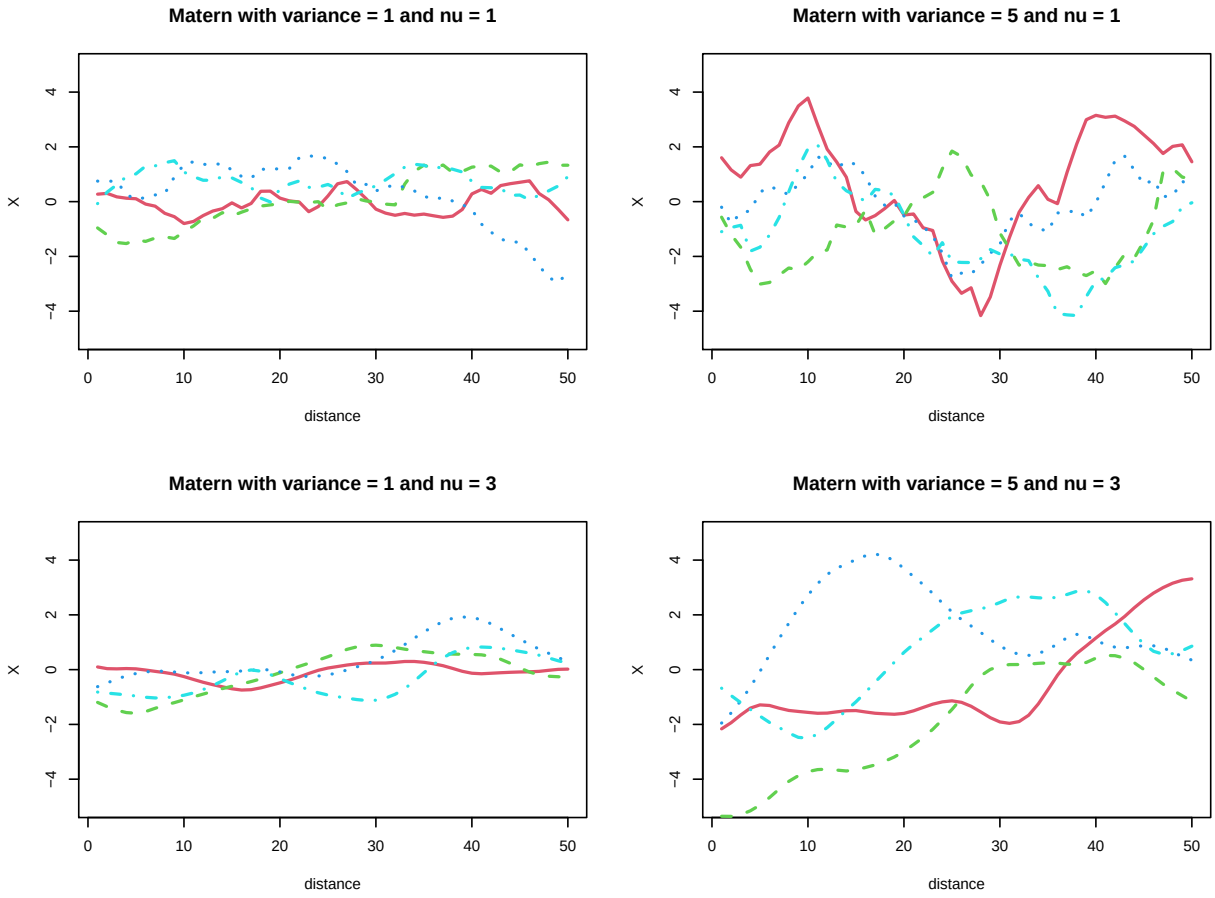
We simulate four realizations from each model.

```
# Create covariance matrices for the powered exponential model

sigma_pwd_exp_1_1 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="powered.exponential"

sigma_pwd_exp_5_1 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="powered.exponential"

sigma_pwd_exp_1_1.9 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="powered.exponentia

sigma_pwd_exp_5_1.9 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="powered.exponentia



# Create covariance matrices for the matern model

sigma_matern_1_1 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="matern", cov.pars=c(1

sigma_matern_5_1 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="matern", cov.pars=c(5

sigma_matern_1_3 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="matern", cov.pars=c(1

sigma_matern_5_3 = cov.spatial(as.matrix(dist(expand.grid(D_domain))), cov.model="matern", cov.pars=c(5



# Create the mean vector

mean_vec = rep(0,50)



# Find realizations of X on the discrete domain

## Realizations using the powered exponential covariance models

real_pwd_exp_1_1 = t(mvrnorm(n=4, mean_vec, sigma_pwd_exp_1_1))

real_pwd_exp_5_1 = t(mvrnorm(n=4, mean_vec, sigma_pwd_exp_5_1))

real_pwd_exp_1_1.9 = t(mvrnorm(n=4, mean_vec, sigma_pwd_exp_1_1.9))
```

```r
real_pwd_exp_5_1.9 = t(mvrnorm(n=4, mean_vec, sigma_pwd_exp_5_1.9))



## Realizations using the Matern covariance models

real_matern_1_1 = t(mvrnorm(n=4, mean_vec, sigma_matern_1_1))

real_matern_5_1 = t(mvrnorm(n=4, mean_vec, sigma_matern_5_1))

real_matern_1_3 = t(mvrnorm(n=4, mean_vec, sigma_matern_1_3))

real_matern_5_3 = t(mvrnorm(n=4, mean_vec, sigma_matern_5_3))



# Plot the realizations of the powered exponential covariance model

par(mfrow = c(2,2), cex = 0.6)

## Powered exponential with variance = 1 and alpha = 1
matplot(as.data.frame(real_pwd_exp_1_1), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste(

## Powered exponential with variance = 5 and alpha = 1
matplot(as.data.frame(real_pwd_exp_5_1), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste(

## Powered exponential with variance = 1 and alpha = 1.9
matplot(as.data.frame(real_pwd_exp_1_1.9), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste

## Powered exponential with variance = 5 and alpha = 1.9
matplot(as.data.frame(real_pwd_exp_5_1.9), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste
```




```
# Plot the realizations of the Matern covariance model

par(mfrow = c(2,2), cex = 0.6)

## Matern with variance = 1 and nu = 1
matplot(as.data.frame(real_matern_1_1), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste('

## Matern with variance = 5 and nu = 1
matplot(as.data.frame(real_matern_5_1), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste('

## Matern with variance = 1 and nu = 3
matplot(as.data.frame(real_matern_1_3), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste('

## Matern with variance = 5 and nu = 3
matplot(as.data.frame(real_matern_5_3), type = 'l', col = c(2:5), ylim=c(-5,5), lwd = 2, main = paste('
```



As we can see, the plots depend both on α (in the powered exponential) or ν (in the Matern model) and σ^2. Increased marginal variance leads to higher variance in the realizations, as expected.

In the model with the Matern covariance function, changing ν often results in changing the (mean squared) differentiability of the process. One can show that the process will be $\lceil \nu \rceil - 1$ times differentiable. Hence, changing ν from 1 to 3 takes the differentiability from zero times differentiable to two times differentiable, as we can anticipate from the plots.

If we would let ν go to infinity, it would approach a Gaussian covariance function (powered exponential with $\alpha = 2$) and for $\nu = 0.5$, we would get the exponential covariance function ($\alpha = 1$).

Hence, α controls the degree of differentiability of the process with powered exponential covariance function, as we can see from the plots. As $\alpha \to 2$, it becomes smooth, while for $\alpha = 1$, it would not even be differentiable.

We also had the parameter $a = 10$ for the powered exponential and $a = 20$ for the Matern. This parameter controls how fast the correlation decays with distance. As we have a higher a for the Matern, we could expect that it fluctuates less over distance, which is also what we observe in the realizations. The powered exponential models (where $a = 10$) go more "up and down", while the Matern models (where $a = 20$) "stays flat" more often.

c)

We set $s_1 = 10, s_2 = 25$ and $s_3 = 30$. At these points, the observations are given by $Y_i = X(s_i) + \varepsilon_i$ for $i = 1, 2, 3$, where $\varepsilon_1, \varepsilon_2, \varepsilon_3 \overset{iid}{\sim} \mathcal{N}(0, \sigma_N^2)$. We also have $X(s_i) \sim \mathcal{N}(0, \sigma^2)$ for each i.

Hence, each Y_i is distributed as $Y_i \sim \mathcal{N}(0, \sigma^2 + \sigma_N^2)$ as the errors ε_i are independent of the $X(s_i)$'s.

This implies that the vector $\mathbf{Y} = (Y_1, Y_2, Y_3)^T$ is distributed as a multivariate normal distribution, $\mathbf{Y} \sim \mathcal{N}_3(0, \Sigma)$, where the covariance matrix Σ is given by

$$\begin{aligned}\Sigma_{ij} &= Cov[Y_i, Y_j] = Cov[X(s_i) + \varepsilon_i, X(s_j) + \varepsilon_j] \\ &= Cov[X(s_i), X(s_j)] + Cov[X(s_i), \varepsilon_j] + Cov[\varepsilon_i, X(s_j)] + Cov[\varepsilon_i, \varepsilon_j] \\ &= \begin{cases} \sigma^2 + \sigma_N^2 & i = j \\ \sigma^2 \rho(\|s_i - s_j\|) & i \neq j. \end{cases}\end{aligned}$$

d)

We consider the Matern covariance model with $\sigma^2 = 5$ and $\nu = 3$ and choose the first realization.

We consider first the vector $[\mathbf{X}, \mathbf{Y}]^T \sim \mathcal{N}([\mu_0, \mu_1]^T, \Sigma)$, where $[\mu_0, \mu_1]^T = [0, 0]^T$ and $\Sigma = \begin{bmatrix} \Sigma_{00} & \Sigma_{01} \\ \Sigma_{10} & \Sigma_{11} \end{bmatrix}$. Here Σ_{00} and Σ_{11} are the covariance matrices of $\mathbf{X}$ and $\mathbf{Y}$ respectively and $\Sigma_{01} = \Sigma_{10}^T = [Cov(X_i, Y_j)]_{ij}$ for $i = 1, \ldots, 50$ and $j = 1, 2, 3$.

We can calculate the covariance as follows,

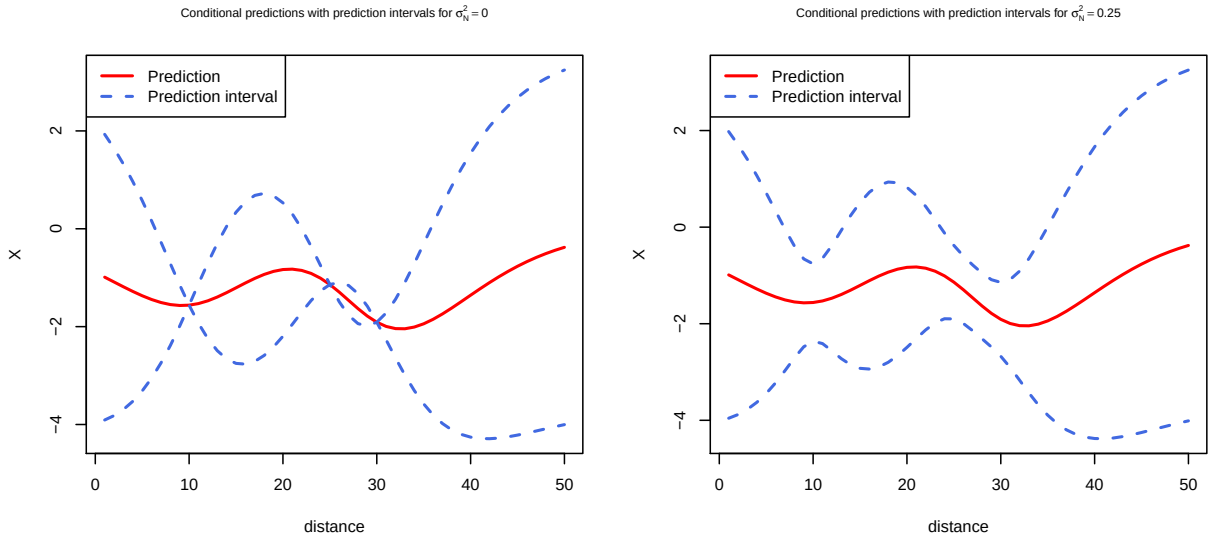
$$Cov(X_i, Y_j) = Cov(X_i, X_j + \varepsilon_j) = Cov(X_i, X_j) + Cov(X_i, \varepsilon_j) = Cov(X_i, X_j).$$

Hence, $\Sigma_{01} = \Sigma_{10}^T = [Cov(X_i, X_j)]_{ij}$ for $i = 1, \ldots, 50$ and $j = 10, 25, 30$.

The conditional distribution can be shown to be given by $\mathbf{X}|_{\mathbf{Y}=y} \sim \mathcal{N}(\mu_0 + \Sigma_{01}\Sigma_{11}^{-1}(y - \mu_1), \Sigma_{00} - \Sigma_{01}\Sigma_{11}^{-1}\Sigma_{10})$.

```
# Variances for the errors
sigma_N = c(0,0.25)

# Name our realization
matern_vec_5_3 = real_matern_5_3[,1]

# Find the values at the selected locations
y = matern_vec_5_3[c(10,25,30)]




#Define covariance matrices

sigma_00 = sigma_matern_5_3

# Define for both possible observation variances

sigma_11_0 <- cov.spatial(as.matrix(dist(expand.grid(c(10,25,30)))), cov.model="matern", cov.pars=c(5,ra

sigma_11_025 <- cov.spatial(as.matrix(dist(expand.grid(c(10,25,30)))), cov.model="matern", cov.pars=c(5

sigma_11_0 <- cov.spatial(as.matrix(dist(expand.grid(c(10,25,30)))), cov.model="matern", cov.pars=c(5,ra

# Find the matrices off-diagonal
sigma_01 = sigma_00[,c(10,25,30)]
```

```r
sigma_10 = t(sigma_01)


# Make mean vector to construct pred. intervals

cond_mean = sigma_01%*%solve(sigma_11_0)%*%y

# Make prediction interval for observation variance 0
upper_PI_0 = cond_mean + sqrt(diag(sigma_00 - sigma_01%*%solve(sigma_11_0)%*%sigma_10))*qnorm(0.05, low

lower_PI_0 = cond_mean - sqrt(diag(sigma_00 - sigma_01%*%solve(sigma_11_0)%*%sigma_10))*qnorm(0.05, low


# Make prediction interval for observation variance 0.25
upper_PI_025 = cond_mean + sqrt(diag(sigma_00 - sigma_01%*%solve(sigma_11_025)%*%sigma_10))*qnorm(0.05,

lower_PI_025 = cond_mean - sqrt(diag(sigma_00 - sigma_01%*%solve(sigma_11_025)%*%sigma_10))*qnorm(0.05,



# Plot the predictions with their prediction intervals

par(mfrow = c(1,2),cex.main = 0.7, cex = 0.7)

## Plot for observation variance 0
matplot(as.data.frame(cbind(cond_mean,upper_PI_0,lower_PI_0)), type = 'l', lty = c(1,2,2), col = c('red

legend('topleft', c("Prediction", "Prediction interval"), lty=c(1,2), col = c('red','royalblue'), lwd =



## Plot for observation variance 0.25

matplot(as.data.frame(cbind(cond_mean,upper_PI_025,lower_PI_025)), type = 'l', lty = c(1,2,2), col = c(

legend('topleft', c("Prediction", "Prediction interval"), lty=c(1,2), col = c('red','royalblue'), lwd =
```



To make the prediction intervals, we used the known mean and variance and the properties of the normal distr. i.e. $\frac{X_{pred}-\mu}{\sigma} \sim \mathcal{N}(0,1)$ to calculate the upper and lower bounds of the prediction intervals, i.e.

$$P(-z_{0.05} \leq \frac{X_{pred}-\mu}{\sigma} \leq z_{0.05}) = 90\% \implies P(X_{pred} \in [\mu - z_{0.05}\sigma, \mu + z_{0.05}\sigma]) = 90\%.$$

We found σ^2 by taking the diagonal of the covariance matrix.

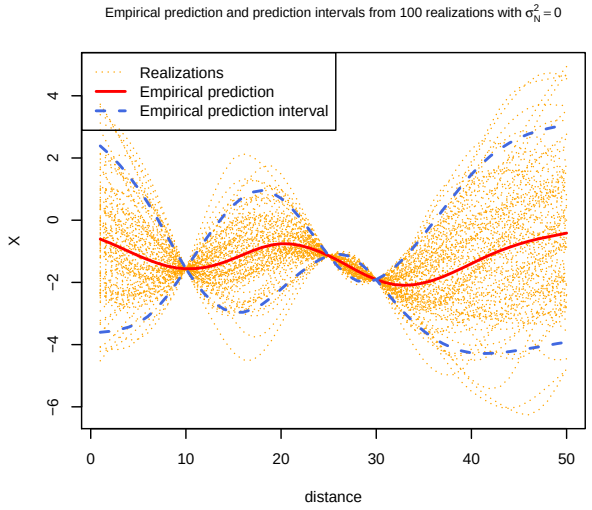
From both plots it is clear that the observation decrease the variance. In the left figure $\sigma_N^2 = 0$, hence there is no uncertainty in the points 10, 25 and 30. The variance is lower at this points in the right plot as well, but since there is some uncertainty in the observations the prediction interval is wider here. At $s = 0$ and $s = 50$ the prediction intervals are the widest, as the observations are too far away to give any accurate indication of the values at the ends.

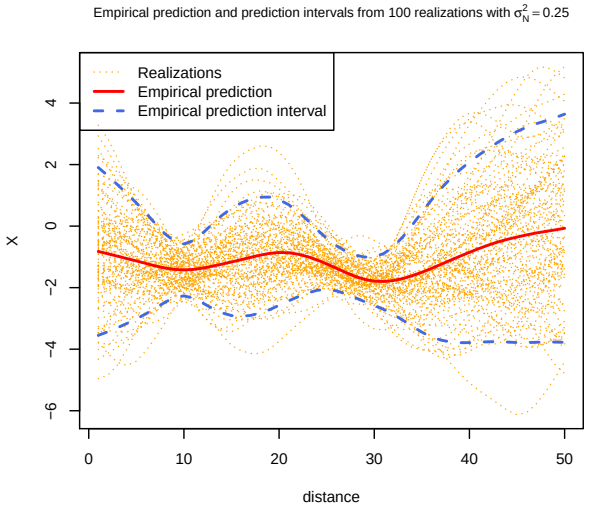
e)

We now compute 100 realizations from the conditional model $\mathbf{X}|_{\mathbf{Y}=y} \sim \mathcal{N}(\mu_0 + \Sigma_{01}\Sigma_{11}^{-1}(y-\mu_1), \Sigma_{00} - \Sigma_{01}\Sigma_{11}^{-1}\Sigma_{10})$, and use these to estimate a prediction of X with prediction intervals.

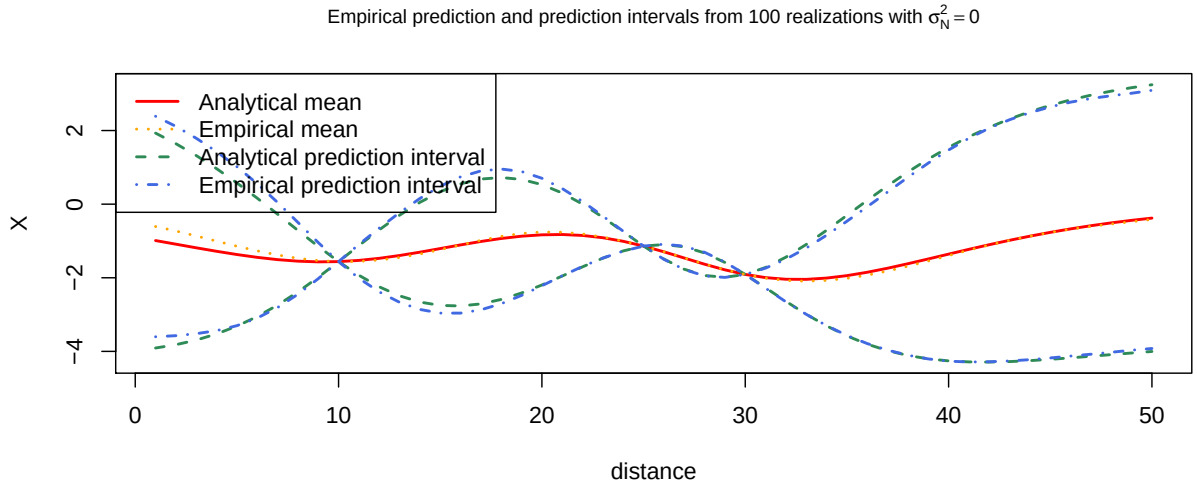
```
# Simulate 100 predictions of the conditional of observation variance 0
cond_sim_100_0 = mvrnorm(100, sigma_01%*%solve(sigma_11_0)%*%y, sigma_00 - sigma_01%*%solve(sigma_11_0)%

# Simulate 100 predictions of the conditional of observation variance 0.25
cond_sim_100_025 = mvrnorm(100, sigma_01%*%solve(sigma_11_025)%*%y, sigma_00 - sigma_01%*%solve(sigma_1


# Predict empirically by taking means (for variances 0 and 0.25)
emp_pred_0 = colMeans(cond_sim_100_0)
emp_pred_025 = colMeans(cond_sim_100_025)


# Predict the variances empirically by (for variances 0 and 0.25)
var_emp_pred_0 = apply(cond_sim_100_0,2,var)
var_emp_pred_025 = apply(cond_sim_100_025,2,var)
```

```r
# Make empirical prediction intervals for observation variance 0
emp_upper_PI_0 = t(emp_pred_0) + sqrt(var_emp_pred_0)*qt(0.05,99, lower.tail = F)
emp_lower_PI_0 = t(emp_pred_0) - sqrt(var_emp_pred_0)*qt(0.05,99, lower.tail = F)



# Make empirical prediction intervals for observation variance 0.25
emp_upper_PI_025 = t(emp_pred_025) + sqrt(var_emp_pred_025)*qt(0.05,99, lower.tail = F)
emp_lower_PI_025 = t(emp_pred_025) - sqrt(var_emp_pred_025)*qt(0.05,99, lower.tail = F)



# Plot the sampled predictions with their empirical prediction intervals
par(mfrow=c(1,2), cex.main = 0.85, cex = 0.7)

## Plot for observation variance 0

# number of realizations in plot
n_0=100

# Plot
matplot(as.data.frame(t(rbind(cond_sim_100_0[1:n_0,],emp_pred_0, emp_upper_PI_0,emp_lower_PI_0))), type

legend('topleft', c("Realizations", "Empirical prediction", "Empirical prediction interval"), lty=c(3,1


## Plot for observation variance 0.25

# number of realizations in plot
n_025=100

# Plot
matplot(as.data.frame(t(rbind(cond_sim_100_025[1:n_025,],emp_pred_025, emp_upper_PI_025,emp_lower_PI_025

legend('topleft', c("Realizations", "Empirical prediction", "Empirical prediction interval"), lty=c(3,1
```






```
par(mfrow = c(1,1))

# Plot for comparing empirical and known values of the mean and PIs
matplot(as.data.frame(cbind(cond_mean,upper_PI_0,lower_PI_0,as.matrix(emp_pred_0), t(emp_upper_PI_0),t(

legend('topleft', c("Analytical mean", "Empirical mean", "Analytical prediction interval", "Empirical pi
```



To make the prediction intervals, we had to calculate the empirical mean and variance. And then we used the property that $\frac{X_{pred}-\bar{X}}{\sqrt{s^2(1+1/n)}} \sim t_{n-1}$ to calculate the upper and lower bounds of the prediction intervals as follows

$$P(-t_{99,0.05} \leq \frac{X_{pred}-\mu}{s} \leq t_{99,0.05}) = 90\% \implies P(X_{pred} \in [\mu - t_{99,0.05}s, \mu + t_{99,0.05}s]) = 90\%,$$

where s is the estimated standard deviation.

The empirical prediction interval should contain approximately 90% of our observations, which seems true. Furthermore, we again see the trend of lower vaiance close to the observed points, and higher variance further away. From the last plot we can compare the empirical and analytical results, and we see that they are pretty consistent. This is not too surprising, as 100 realizaitons is a quite good sample size. In general the empirical prediction interval is wider, which is what we would expect. However, from the last observation and out, i.e. for $s \in [30, 50]$, the empricial prediction interval is actually smaller. This might be a result of "lucky" realizations, where less than expected are outside the analytical prediction interval, hence underestimating the variance.

f)

We now want to estimate the expected size of the area where $X > 2$ in two ways. First, let $A = \sum_{s \in \tilde{\mathcal{D}}} \mathbb{I}(X(s) > 2)(X(s) - 2)$ be a function of $X(s)$. One way to find the expected size of this area is to calculate $E[A(X(s))] = \hat{A}$.

Another method is to calculate $\tilde{A} = A(\hat{X})$, where $\hat{X}$ is the simple kriging predictor of X. The simple Kriging predictor at a location s_0 is given by $\hat{X}_0 = c^T \Sigma^{-1} X$, where $X = (X(s_1), \ldots, X(s_n)$, $c = Cov(X, X(s_0))$ and $\Sigma = Cov(X, X)$.

```
# We make an indicator function for x > 2
ind <- function(x){x>2}
A <- function(v){          # takes in a vector
  return(sum(sapply(v, ind)*(v-2)))
}

# Calculate A-hat by calculating A for each realization and then finding the mean and variance

A_hat = mean(apply(cond_sim_100_0,2,A))
A_hat_var = var(apply(cond_sim_100_0,2,A))

# Calculate A-tilde using the simple kriging predictors

## Define the observation locations to construct the Sigma matrix.
obs_locs = c(10,25,30)

## Construct the matrices needed for the simple Kriging prediction.
simp_krig_sigma = cov.spatial(as.matrix(dist(expand.grid(obs_locs))), cov.model="matern", cov.pars=c(5,1

simp_krig_c = sigma_matern_5_3[obs_locs,]

# Kriging!
simp_krig_pred = t(sigma_10)%*%solve(sigma_11_0)%*%(y)

# Find the predictor of the area above 2 as earlier.
A_tilde = A(simp_krig_pred)

area_A = matrix(c(round(A_tilde,3),round(A_hat,3), round(A_hat_var,3)), nrow = 1)
colnames(area_A) = c('Ã', 'Â', 'Var(Â)')
area_A
```

```
##      Ã     Â Var(Â)
## [1,] 0 2.213 20.052
```

We see that $\tilde{A}$ predicts 0 wheras $\hat{A}$ predicts more than 1. This is likely due to the fact that $\hat{A}$ has multiple realizations that passes 2 due to high variance, and this allows for an estimate $\hat{A} > 0$. When using simple kriging we have already found the expected value of X in each point and in no point is our GRF expected to pass 2. The high variance of $\hat{A}$ show that both estimates agree that it is reasonable for X to never pass 2, i.e. that the area is 0. That $\tilde{A} \leq \hat{A}$ will always be true, and can be shown as follows.

Notice that A is a convex function. Additionally, since X is a GRF we know that the simple kriging predictor of X is also the expectation of X in that point, i.e. $\hat{X} = E[X]$.

Using Jensen's inequality, which says that

$$g(E[X]) \leq E[g(X)],$$

it follows immediately that $\tilde{A} \leq \hat{A}$. To see this, note that

$$\tilde{A} = A(\hat{X}) = A(E[X]) \leq E[A(X)] = \hat{A},$$

by Jensen's inequality.

g)

In task one we have seen that the parameters ν for Matérn and α for powered exponential determines the smoothness of our covariance function, whereas the marginal variance σ^2 and range a determines the variance of our function. A high marginal variance σ^2 can be "held in check" if the range too is high. Thus, the potential variance is high but a realization of the RF will change slowly since the range restricts it from behaving very unpredictable.

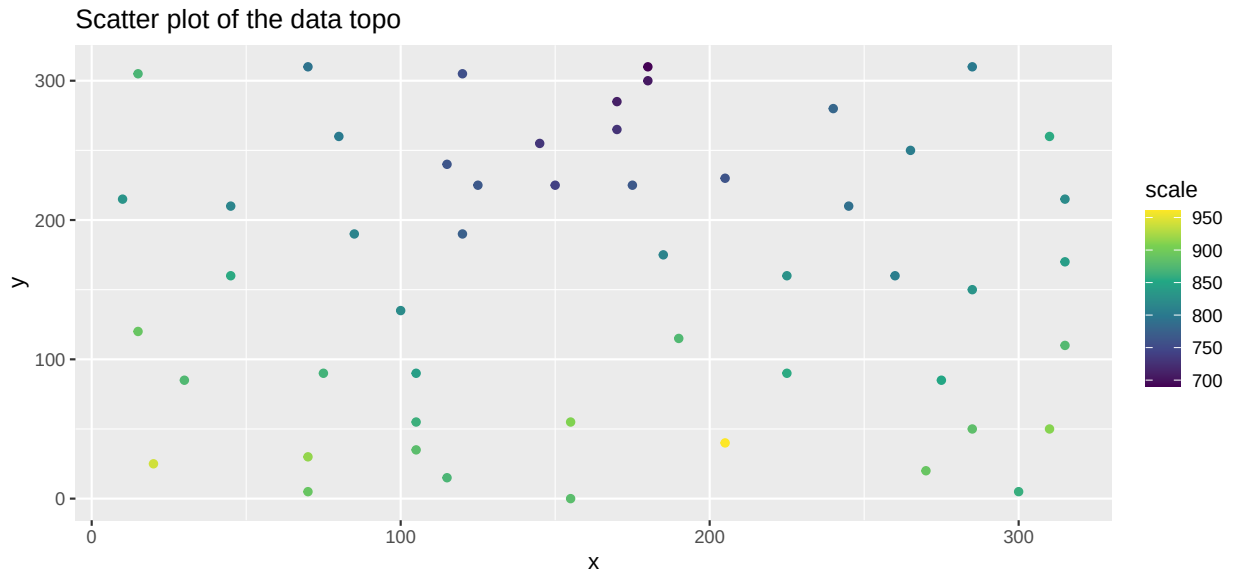
Furthermore, we noticed that observations can significantly reduce variance of our predictions. However, when the range is small and the variance is high, if we only observe a few data points we should not be overconfident in estimates far from our observations.

Lastly, we saw that if we want to calculate something from our observations we have to be careful of whether we choose to find a prediction based on our observations and then calculate what we are interested in, or if we first choose to calculate what we want based on all observations and then predict our calculation from this. These two methods will not yield the same answer.

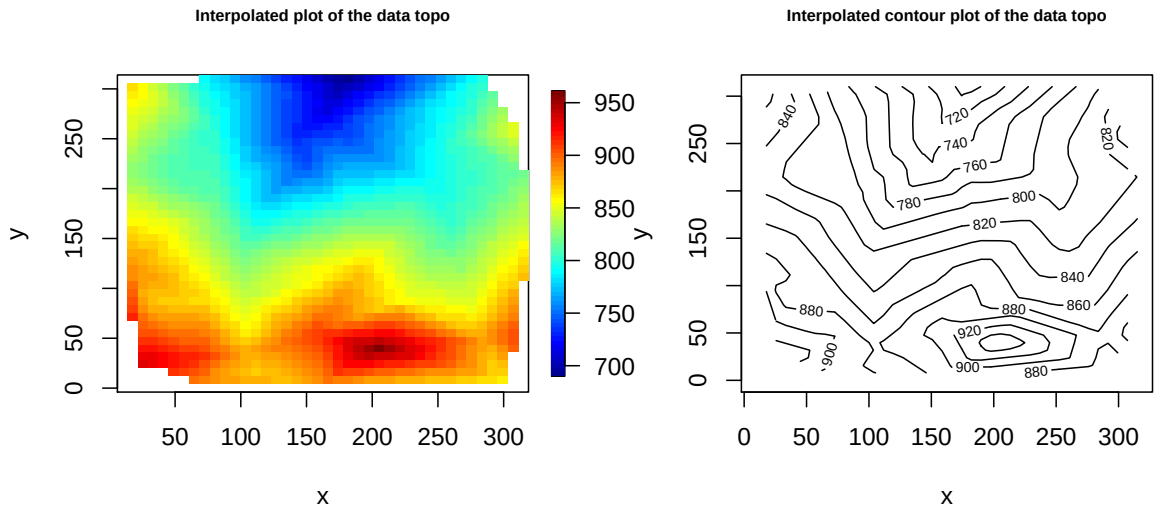
Problem 2

a)

```
# read in the datafile topo.dat
df_topo <- read.delim('topo.dat', sep = ' ')


# plot the points with color as a indicator of the value in each coordinate
ggplot(data = df_topo) + geom_point(aes(x,y,color = z)) +
  scale_color_viridis_c(name="scale") + ggtitle("Scatter plot of the data topo")
```




```
par(mfrow = c(1,2), cex.main=0.7)
# plot an interpolated plot to get an idea of the pattern in non-observed points
image.plot(interp(df_topo$x,df_topo$y,df_topo$z), xlab = 'x', ylab = 'y', main = 'Interpolated plot of t

# plot an interpolated contour plot to get an idea of the pattern in non-observed points
contour(interp(df_topo$x,df_topo$y,df_topo$z), xlab = 'x', ylab = 'y', main = 'Interpolated contour plot
```



From the plots we would not assume that this is a stationary GRF since there is no clear consistency in the covariance over certain distances over the different areas in the plot. For instance, the upper region of the plot seem to have more dependency than the lower region. In the scatter plot as well we see that the yellow point above 200 on the x-axis seem very odd, and not displaying the strong dependence to its neighbouring points as seen for the dots in the upper region.

b)

We now assume that the GRF X on the domain $\mathcal{D} \in [0, 350] \times [0, 350] \subset \mathbb{R}^2$ is modeled by

$$E[X(s)] = g(s)^T\beta, \quad s \in \mathcal{D} Var[X(s)] = \sigma^2 = 50^2, \quad s \in \mathcal{D} Corr[X(s), X(s')] = \rho(\|s-s'\|) \overset{h=\|s-s'\|}{=} e^{-(0.01h)^{1.5}}, \quad h \in [0, \infty),$$

We recognize the correlation function as a powered exponential.

If we want to predict an unobserved value of X at the point s_0, we could use universal kriging. Universal kriging finds the BLUP, i.e. the best liner unbiased estimate $\hat{X}_0$ of $X(s_0) = X_0$. This means we are looking for $\hat{X}_0$ satisfying the following:

- **Best:** The mean square error $E[(X_0 - \hat{X}_0)^2]$ is minimized.
- **Unbiased:** $E[\hat{X}_0] = E[X_0]$
- **Linear:** $\hat{X}_0 = \mathbf{a}^T\mathbf{X}$, i.e. it is a linear combination of our observations.

From this we see that we get a constraint on β from

$$\begin{aligned} E[\hat{X}_0] &= E[X_0] \\ E[\mathbf{a}^T\mathbf{X}] &= E[X_0] \\ \mathbf{a}^T Z^T\beta &= Z_0^T\beta \\ \mathbf{a}^T Z^T\beta - Z_0^T\beta &= 0 \\ (\mathbf{a}^T Z^T - Z_0^T)\beta &= 0 \end{aligned}$$

where the mean function $m(s) = \sum_{k=1}^{p} f_k(s)\beta_k = Z_s^T\beta$ and $Z = (Z_1, \ldots, Z_n)$.

From this we can minimize the MSE using Lagrange multipliers to incorporate the constraint on beta by solving

$$\min_{\beta} \left(E[(Z_0^T\beta - \mathbf{a}^T Z^T\beta)^2] - \lambda((\mathbf{a}^T Z^T - Z_0^T)\beta) \right)$$

The prediction variance σ_{uk}^2 is given by

$$\begin{aligned} \sigma_{uk}^2 &= MSE(\hat{\beta}) \\ &= E[(Z_0^T\hat{\beta} - \mathbf{a}^T Z^T\hat{\beta})^2] \\ &= \hat{\beta}^T Var(Z_0)\hat{\beta} + (\mathbf{a}\hat{\beta})^T Var(Z)\mathbf{a}\hat{\beta} - 2Cov(Z_0^T\hat{\beta}, \mathbf{a}^T Z^T\hat{\beta}) \\ &= \hat{\beta}^T Var(Z_0)\hat{\beta} + (\mathbf{a}\hat{\beta})^T Var(Z)\mathbf{a}\hat{\beta} - 2\hat{\beta}^T Cov(Z_0, Z)\mathbf{a}\hat{\beta} \end{aligned}$$

We would expect that the value of σ^2 change for different parametrizations of the expectation function, as less fluctuations of the mean function (i.e. constant or 1st degree linearity) will potentially require more variance in order to explain all the observations, whereas higher degree mean functions may be biased against the observations and therefore explain most of the data without too much variance.

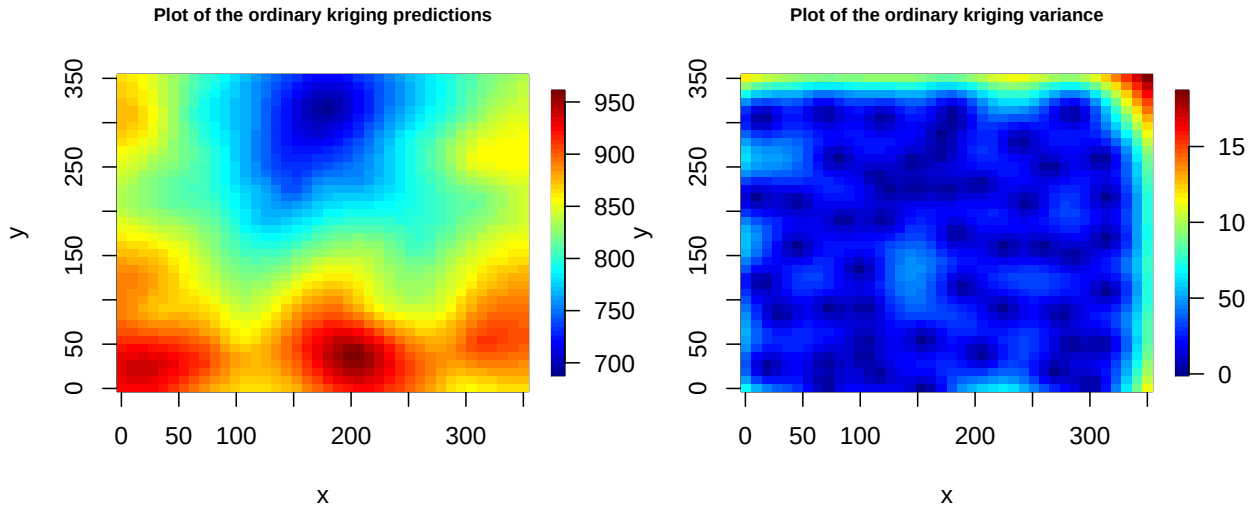
c)

We now assume $E[\mathbf{X}(s)] = \beta_1$, and we want to predict using ordinary Kriging.

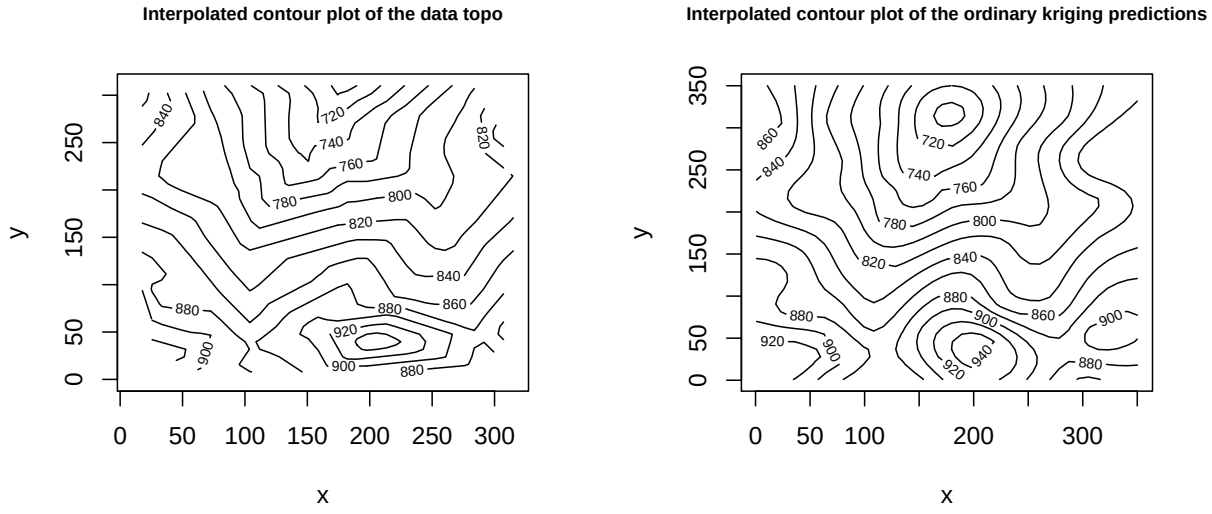
```r
# we wish to use ordinary kriging on the dataset topo
geo_topo <- as.geodata(df_topo) # make our dataset into geodata

# For computational reasons we use a more discreet domain
fine_domain = seq(1,350,1)
disc_domain = seq(5,350,5)

# make our domain as a grid
topo_locations <- expand.grid(fine_domain,fine_domain)

# Use ordinary kriging with constant trend and covariance parameters as specified in task
krigPred <- krige.conv(geo_topo,locations = topo_locations,
                       krige = krige.control(type.krige = "ok",
                                             trend.d = "cte",
                                             trend.l = "cte",
                                             cov.pars = c(2500, 100),
                                             cov.model = "powered.exponential",
                                             kappa = 1.5))
```

```
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

```r
# Store the predictions in a vector
topo_ok_prediction <- krigPred$predict

# store the kriging variance in a vector
topo_ok_variance <- krigPred$krige.var

par(mfrow = c(1,2), cex.main = 0.75)

# plot the kriging predictions on our grid
image.plot(interp(topo_locations$Var1,topo_locations$Var2,topo_ok_prediction),xlab = 'x', ylab = 'y', ma

# plot the kriging variance on our grid
image.plot(interp(topo_locations$Var1,topo_locations$Var2,topo_ok_variance), xlab = 'x', ylab = 'y', mai
```




```
# compare the contour plot of our data and our kriging predictions
contour(interp(df_topo$x,df_topo$y,df_topo$z), xlab = 'x', ylab = 'y', main = 'Interpolated contour plo
contour(interp(topo_locations$Var1,topo_locations$Var2,topo_ok_prediction), xlab = 'x', ylab = 'y', mai
```



We see that there in general is low variance for our predictors. It's quite impressive that the estimate is this certain from merely 52 observations. We also see that compared with the interpolated contour plot to the left, the interpolation of the ordinary kriging predictions appear more stationary. This is due to the fact that it is an assumption of the model.

d)

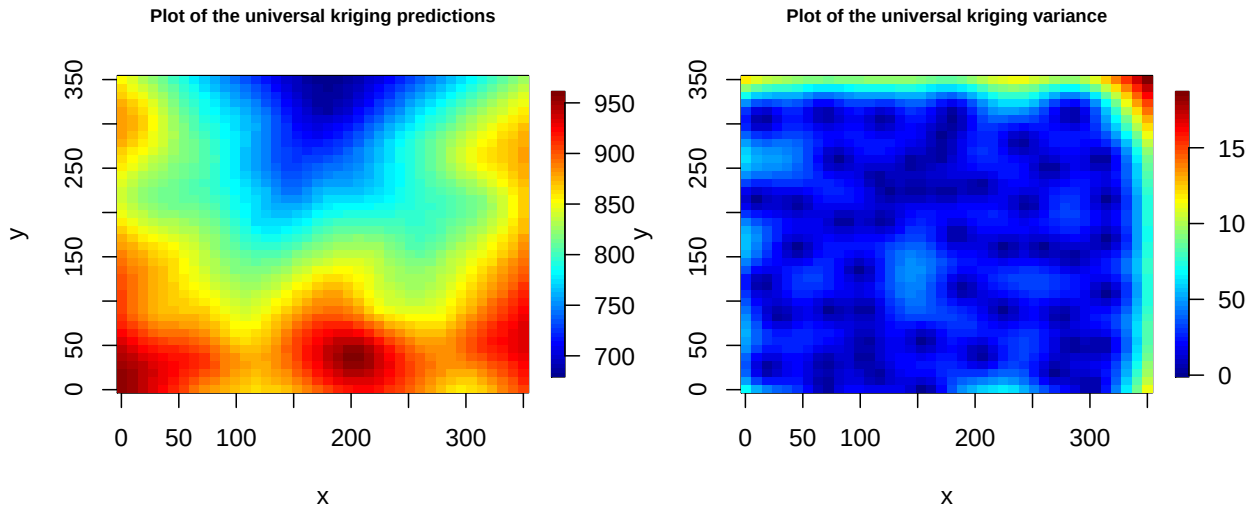
We now let $g(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2)$. The expected value of $\mathbf{X}(s)$ will then be

$$E[\mathbf{X}(s)] = m(s) = \beta_1 + \beta_2 s_1 + \beta_3 s_2 + \beta_4 s_1 s_2 + \beta_5 s_1^2 + \beta_6 s_2^2$$

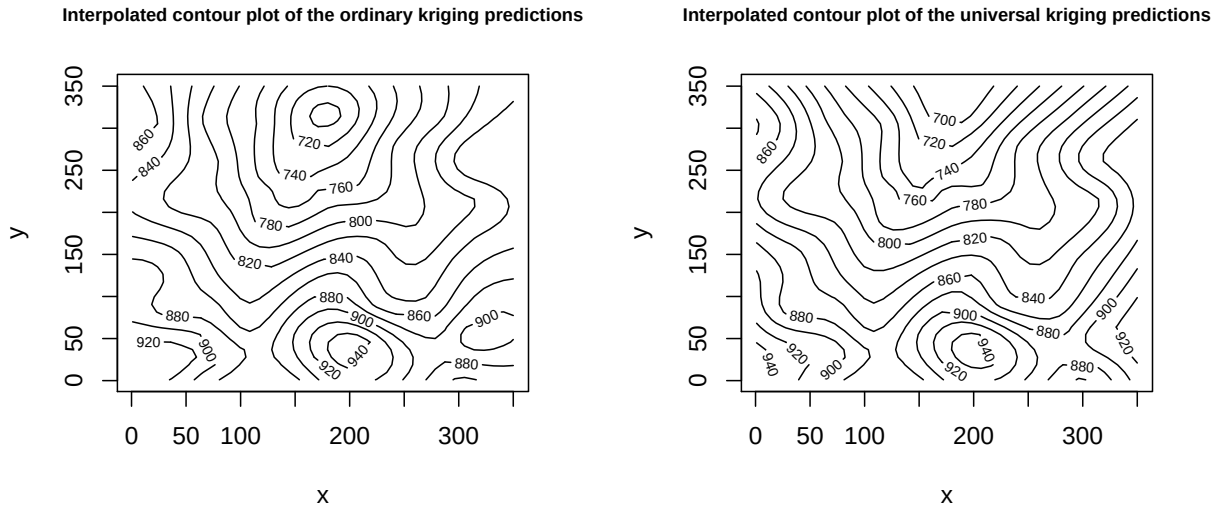
```
# Use universal kriging with covariates g(s) and covariance parameters as specified in task
uk_krigPred <- krige.conv(geo_topo,locations = topo_locations,
                          krige = krige.control(type.krige = "ok",
                                                trend.d = "2nd",
                                                trend.l = "2nd",
                                                cov.pars = c(2500, 100),
                                                cov.model = "powered.exponential",
                                                kappa = 1.5))
```

```
## krige.conv: model with mean given by a 2nd order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```

```
# Store the predictions in a vector
topo_uk_prediction <- uk_krigPred$predict

# store the kriging variance in a vector
topo_uk_variance <- krigPred$krige.var

par(mfrow = c(1,2), cex.main = 0.75)

# plot the kriging predictions on our grid
image.plot(interp(topo_locations$Var1,topo_locations$Var2,topo_uk_prediction),xlab = 'x', ylab = 'y', ma

# plot the kriging variance on our grid
image.plot(interp(topo_locations$Var1,topo_locations$Var2,topo_uk_variance), xlab = 'x', ylab = 'y', mai
```




```
# plot contour plots comparing ordinary and universal kriging
contour(interp(topo_locations$Var1,topo_locations$Var2,topo_ok_prediction), xlab = 'x', ylab = 'y', mai
contour(interp(topo_locations$Var1,topo_locations$Var2,topo_uk_prediction), xlab = 'x', ylab = 'y', mai
```



Again we see that there is quite low variance in our result. An interesting thing to note when we compare universal kriging predictions with the ordinary kriging predictions is that in areas where we have a high value, ordinary kriging does not predict higher values an it also predicts circular height curves around each high value. This is due to the assumption of constant mean. The universal kriging predictions, on the other hand, has less "stationary-looking" predictions, because the polynomial in the mean allows for more fluctuations. We see it in particular in the middle upper region, and in the bottom left and right corner.

e)

We now consider $s_0 = (100, 100)^T$. Our ordinary kriging estimate at this point is 838.6781414. We can use this prediction along with the prediction variance from our ordinary kriging to compute

$$P(X(s_0) > 850) = P\left(\frac{X(s_0) - \hat{X}_0}{\sigma_{ok}} > \frac{850 - \hat{X}_0}{\sigma_{ok}}\right),$$

which we can solve using the quantiles of a student-t distribution.

```
# Extract the prediction and variance in point [100,100]
M = as.data.frame(cbind(topo_locations$Var1,topo_locations$Var2,topo_ok_prediction))
N = as.data.frame(cbind(topo_locations$Var1,topo_locations$Var2,topo_ok_variance))
w_M = which(M$V1==100 & M$V2==100)
w_N = which(N$V1==100 & N$V2==100)


# Standardize and calculate the probability
s0_prob = pt((850-M[w_M,3])/sqrt(N[w_N,3]),51, lower.tail = F)
```

From the calculations we see that $P(X(s_0) > 850) = 13.25\ \%$.

We now want to find the elevation level at which we are 90% sure that the true value $X(s_0)$ is below, i.e. we want to solve $P(X(s_0) < y) = 0.9$ for y. We can rewrite this as $y = t_{51,0.1}\sqrt{\sigma_{ok}} + \hat{X}_0$.

```
# Find the bound by solving for y
s0_bound = qt(0.9,51)*sqrt(N[w_N,3])+M[w_M,3]
```

From the calculations we see that 851.72 is the elevation level at which the true value $X(s_0)$ is 90% certainly below.

We see that the upper bound is very close to the predicted value in this point, again indicating quite low variance.

f)

In this task we used a dataset of 52 observations. Both the ordinary kriging and the universal kriging predictors had low variance, but for this problem we would assume that universal kriging is a better method. This is due to the fact that our data didn't seem certainly stationary, and the polynomial in our mean cut some slack on this restriction. We were surprised by how precise the predictions are from merely 52 observations on a grid of size $[1, 315] \times [1, 315]$.

Problem 3 - Parameter estimation

a)

We now consider a stationary GRF $\{X(s)|s \in \mathcal{D}\}$, where $\mathcal{D} = [1, 30]^2$, with

$$E[X(s)] = \mu = 0 Var[X(s)] = \sigma^2 = 2 Corr[X(s), X(s')] = \exp\left(\frac{-\|s - s'\|}{a}\right) \overset{h=\|s-s'\|}{=} \exp\left(\frac{-h}{3}\right)$$

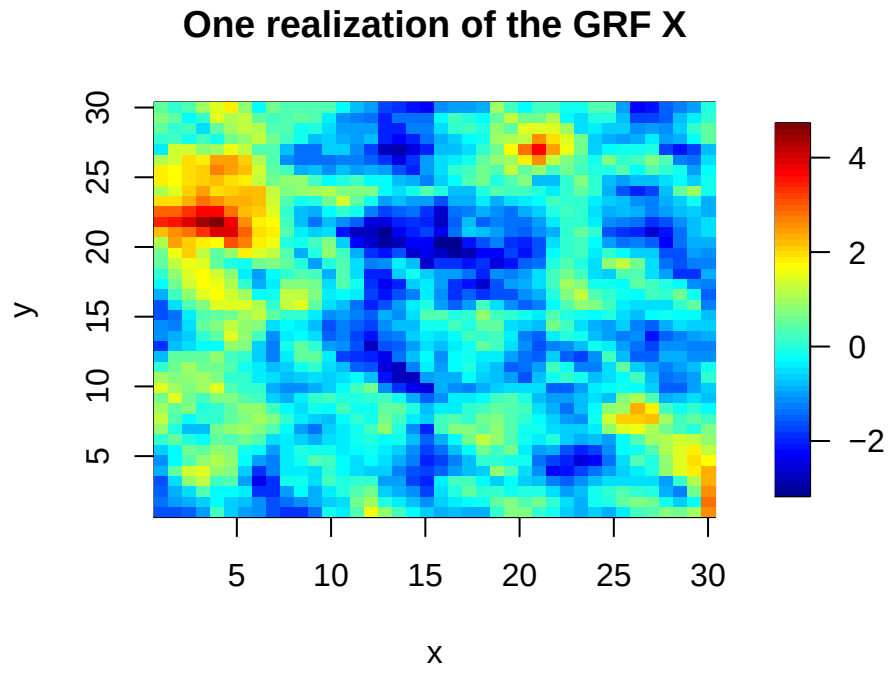
We recognize the correlation function as a powered exponential with $\alpha = 1$ and spatial scale $a = 3$ and find a realization of the GRF.

```
# Make the setup
D_grid = 1:30
sigm = 2
spat_scal = 3

# Construct the covariance matrix

cov_mat_3 = cov.spatial(as.matrix(dist(expand.grid(D_grid))),
                        cov.model="powered.exponential",
                        cov.pars=c(sigm,spat_scal),
                        kappa=1)

# Find a sample using the grf-function.

sample_1 = grf(1,grid = as.matrix(expand.grid(D_grid,D_grid)),
                cov.model="powered.exponential",
                cov.pars=c(sigm,spat_scal),
                        kappa=1)
```

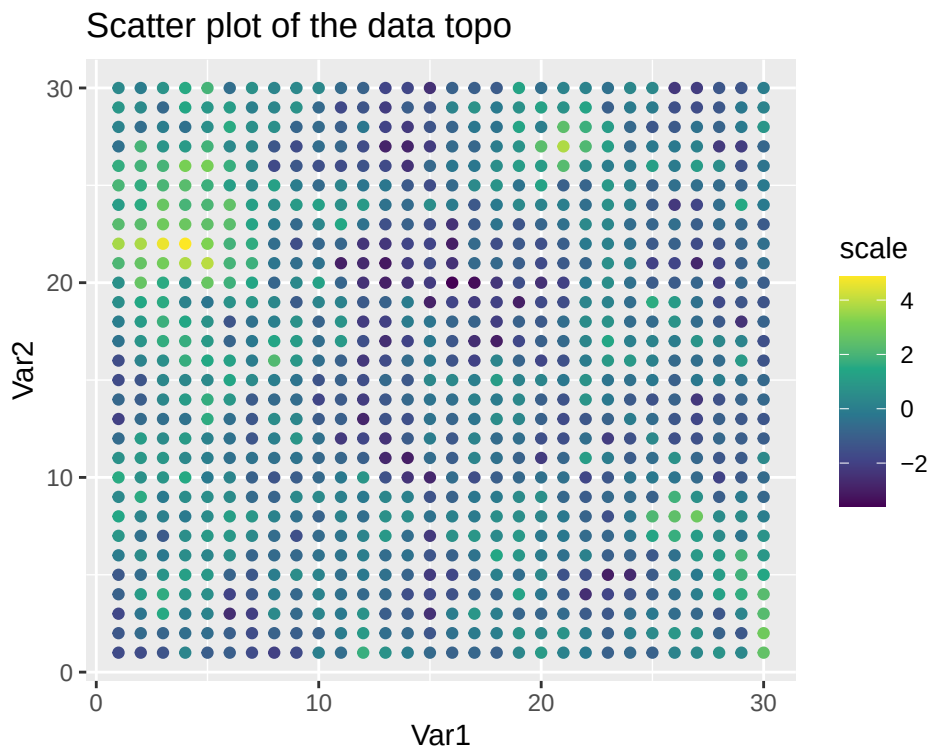
```
## grf: simulation on a set of locations provided by the user
## grf: process with  1  covariance structure(s)
## grf: nugget effect is: tausq= 0
```

```
## grf: covariance model 1 is: powered.exponential(sigmasq=2, phi=3, kappa = 1)
## grf: simulation using the function GaussRF from package RandomFields
## grf: End of simulation procedure. Number of realizations: 1
```

```
# Make it into a df

sample_1_df = as.data.frame(cbind(sample_1$coord, z = sample_1$data))

# Visualize the sample on the grid

## First as a interpolated color plot.
image.plot(interp(sample_1_df$Var1,sample_1_df$Var2, sample_1_df$z), xlab = 'x', ylab = 'y', main = 'One
```




```
## Also as points in each coordinate
ggplot(data = sample_1_df) + geom_point(aes(Var1,Var2,color = z)) +
  scale_color_viridis_c(name="scale") + ggtitle("Scatter plot of the data topo")
```



b)

We now compute the semi-variogram of our GRF, both empirical and the analytical.

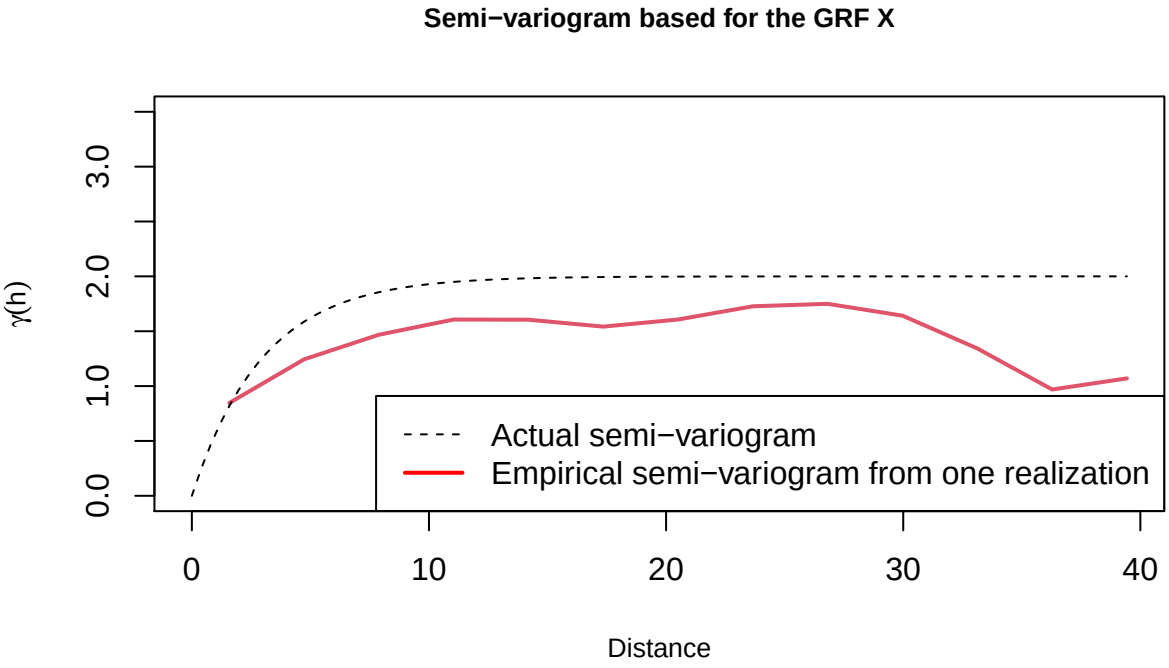
```
par(mfrow = c(2,1),cex.main = 0.8, cex.lab = 0.8)

# empirical and theoretical semi-variograms

plot(sample_1, col = 2, lwd= 2, type = 'l', ylim = c(0,3.5), xlab = 'Distance', ylab = expression(gamma
```

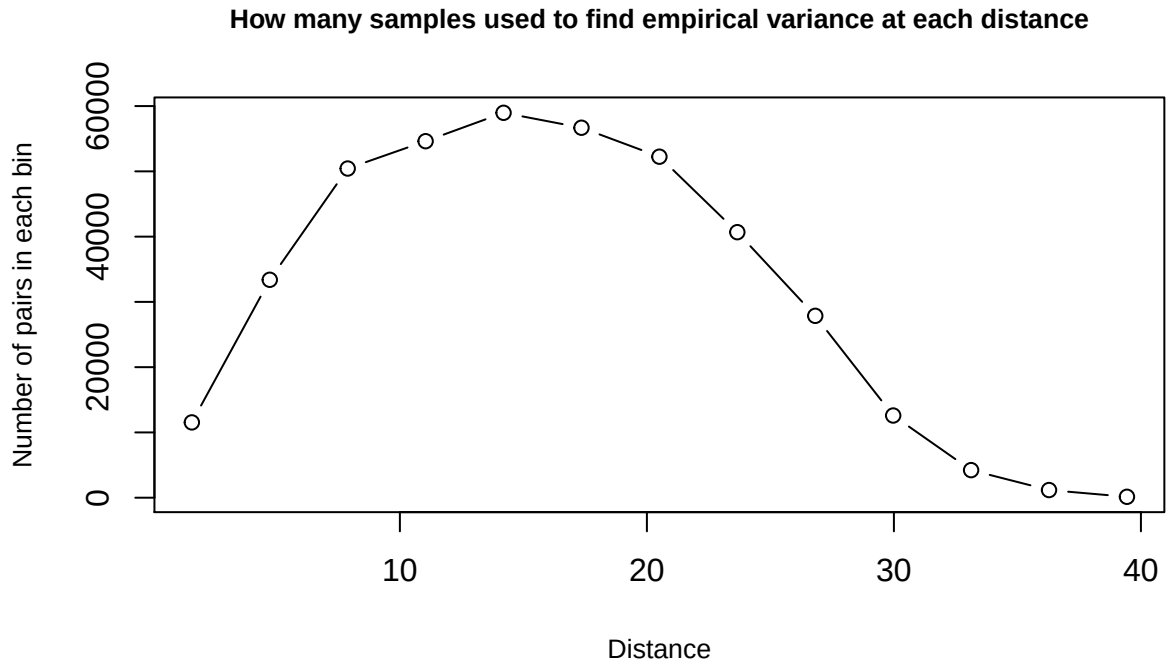
```
## variog: computing omnidirectional variogram
```

```
legend('bottomright', c("Actual semi-variogram", "Empirical semi-variogram from one realization"), lty=
```

```
plot(variog(sample_1)$u,variog(sample_1)$n, type = 'b', xlab = 'Distance', ylab = 'Number of pairs in e
```

```
## variog: computing omnidirectional variogram
## variog: computing omnidirectional variogram
```





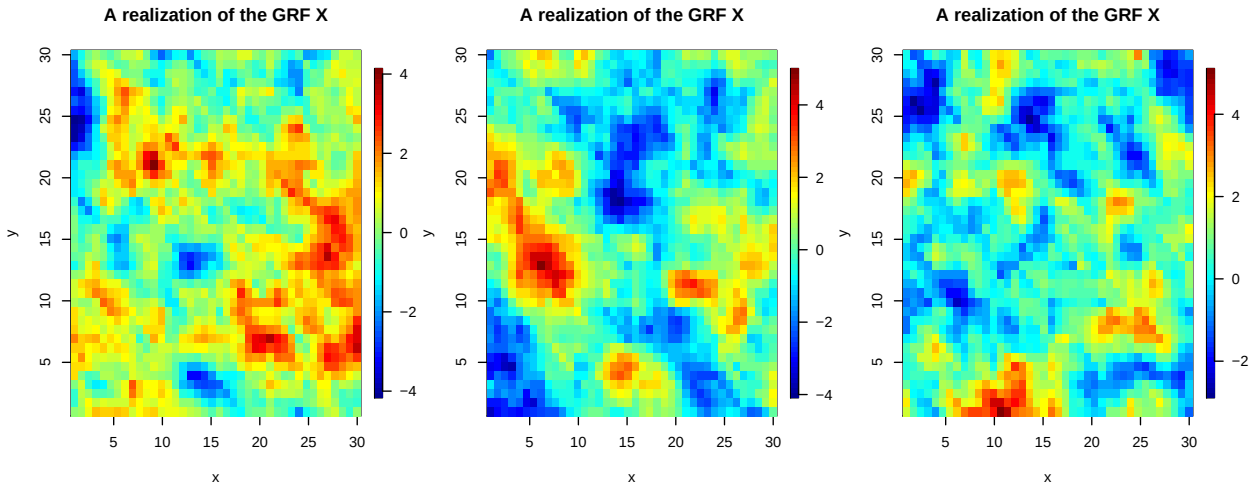
As expected, the empirical semi-variogram is more accurate for lower distances, as there are more samples to compute from. When $\mathcal{D}$ is bounded there are fewer points that are at the distance $h \to diam(\mathcal{D})$ which in this case is approximately 40. We see also from the second plot that there is significant decrease in the amount of pairs to calculate the empirical semi-variogram from as $h \to 40$, which explains why our empirical semi-variogram is inaccurate at the end.

c)

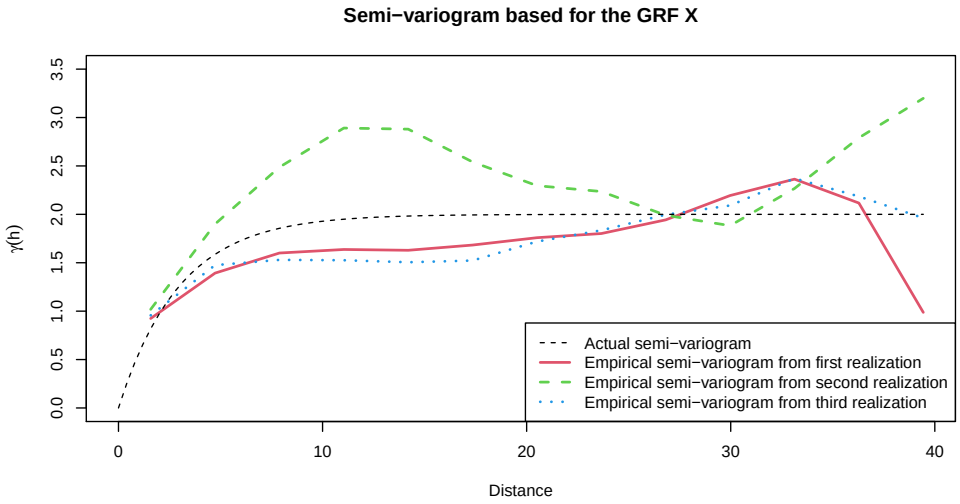
```
sample_3 = grf(1,grid = as.matrix(expand.grid(D_grid,D_grid)), nsim = 3,
               cov.model="powered.exponential",
               cov.pars=c(sigm,spat_scal),
                       kappa=1)
```

```
## grf: simulation on a set of locations provided by the user
## grf: process with  1  covariance structure(s)
## grf: nugget effect is: tausq= 0
## grf: covariance model 1 is: powered.exponential(sigmasq=2, phi=3, kappa = 1)
## grf: simulation using the function GaussRF from package RandomFields
## ...
## grf: End of simulation procedure. Number of realizations: 3
```

```
# Make it into a df

sample_3_df = as.data.frame(cbind(sample_3$coord, z = sample_3$data))

colnames(sample_3_df) = c("Var1", "Var2", "sims1", "sims2", "sims3")

# Visualize the samples on the grids

par(mfrow = c(1,3))

## First as a interpolated color plot.
image.plot(interp(sample_3_df$Var1,sample_3_df$Var2, sample_3_df$sims1), xlab = 'x', ylab = 'y', main =

image.plot(interp(sample_3_df$Var1,sample_3_df$Var2, sample_3_df$sims2), xlab = 'x', ylab = 'y', main =

image.plot(interp(sample_3_df$Var1,sample_3_df$Var2, sample_3_df$sims3), xlab = 'x', ylab = 'y', main =
```



```
par(mfrow = c(1,1), cex = 0.75)

# empirical and theoretical semi-variograms

plot(sample_3, col = c(2,3,4), lwd= 2, type = 'l', ylim = c(0,3.5), xlab = 'Distance', ylab = expressio

## variog: computing omnidirectional variogram

legend('bottomright', c("Actual semi-variogram", "Empirical semi-variogram from first realization", "Em
```



As we can see from the results, we obtain empirical semi-variograms that look similar to the actual semi-variogram. Some of them, such as the ones for the first and third realization, would seem to have a lower sill than the actual variogram, but this is just a random effect. We can see in the image plots of the GRF realizations that the variance in the GRFs seem to correspond to the values in the semi-variogram. For example, one can observe that the second empirical semi-variogram takes a large value around distance 15, and in the corresponding plot of the realization it varies more at this range than the two other plots.

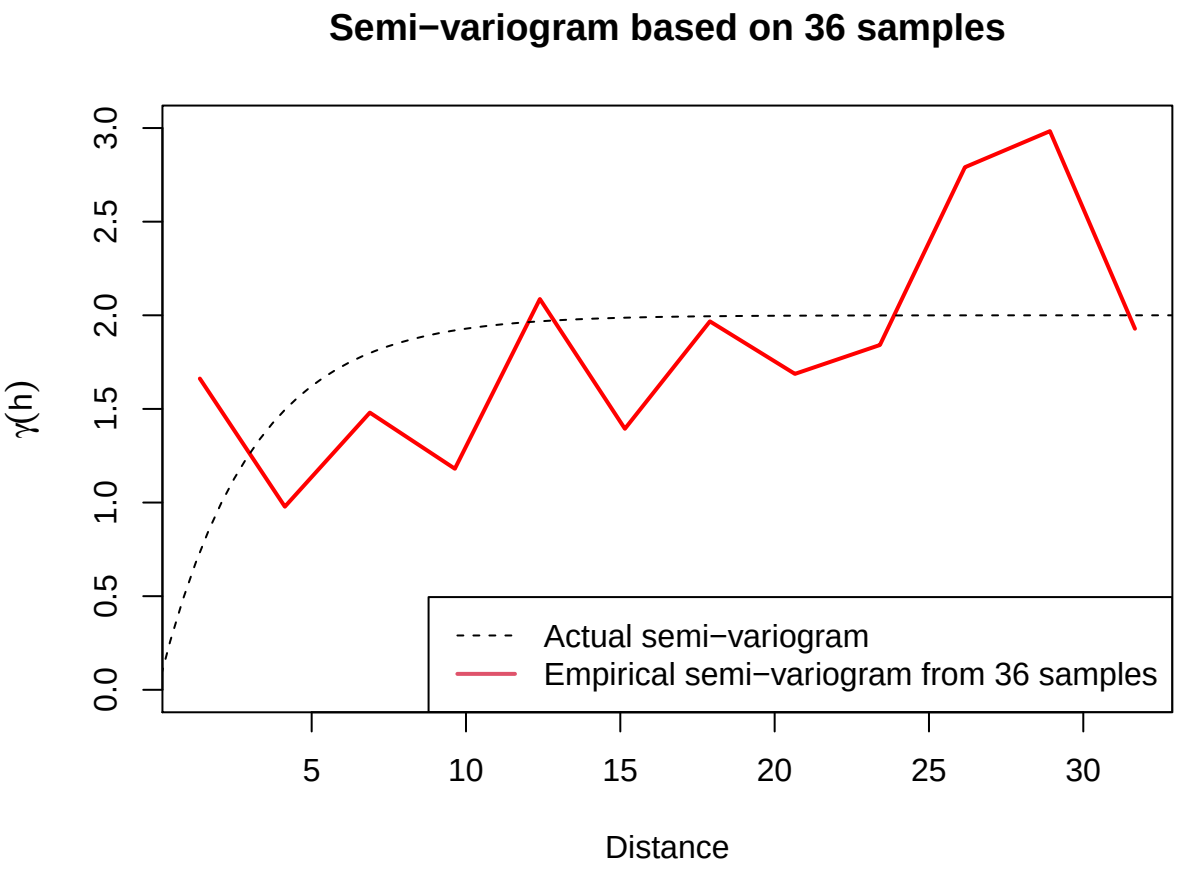
The fluctuations at large distances are caused by having fewer samples at the extreme distances, as shown in the last problem.

d)

We now compare how accurate empirical estimates are for a smaller sample size. We look both at the empirical semi-variogram, as well as an enstimate of the model and the maximum likelihood estimates for the covariance parameters and other eventual parameters. In our case we would expect there to be only the covariance parameters, as the exponential model doesn't take other inputs.

```
# Define the 36 locations and use the first sample (from 3a and 3b) as observations
X = expand.grid(D_grid,D_grid)

locs_36 = sample_1_df[sample(nrow(X),size=36,replace=F),]

# Plot the variogram we observed
plot(variog(as.geodata(locs_36))$u,variog(as.geodata(locs_36))$v, ylim = c(0,3), type = 'l', lty = 1, l
```

```
lines.variomodel(sample_1, lty = 2)

legend('bottomright', c("Actual semi-variogram", "Empirical semi-variogram from 36 samples"), lty=c(2,1
```




```
# Assume unknown model parameters and estimate for all and on the 36 locs
# choose ini.cov.pars based on empirical semi-variogram

# We would guess 2 and 3 as the ini.cov.pars based on the semi-variogram in task b) when using all the
est_all = likfit(as.geodata(sample_1_df), ini.cov.pars = c(2,3))

# We would guess 2.5 and 2 as the ini.cov.pars based on the semi-variogram for only 36 samples
est_36 = likfit(as.geodata(locs_36), ini.cov.pars = c(2.5,2))

est_all_cov_parm = est_all$cov.pars
est_all_cov_mod = est_all$cov.model

est_36_cov_parm = est_36$cov.pars
est_36_cov_mod = est_36$cov.model
```

We first plot the semi-variogram, and then inspected this before choosing initial values in the likfit function. The sill should be the approximately the decay of the empirical semi-variogram, and we guessed the range based on where it looks like most of the dependency is gone. This was quite hard when we only had 36

samples. The true values are $\sigma^2 = 2$ and $a = 3$. Our estimate when using all the locations with initial values $(2, 3)$ was 1.5581282, 2.5158625, respectively, and when we used only the 36 uniformly selected locations with initial parameters $(2.5, 5)$, we obtained 0.9131814, 3.4568497, respectively.

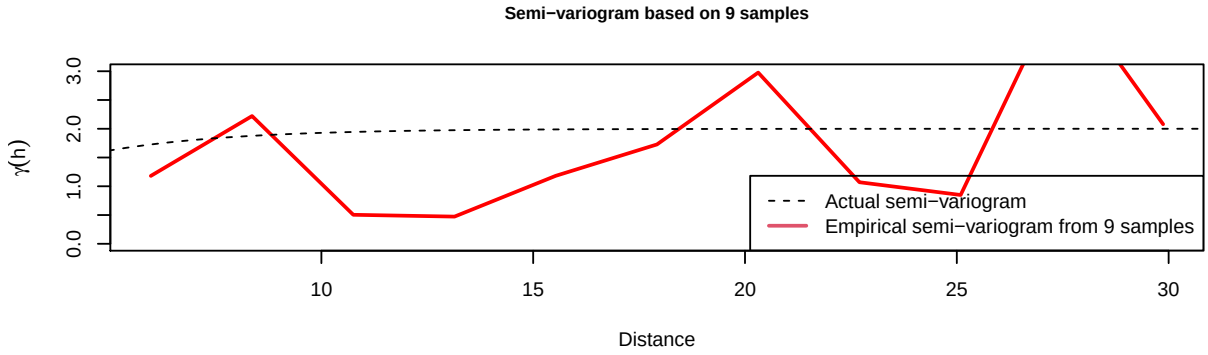
Both likelihood based fits predicted this to be an exponential model, which is correct. If we chose worse initial parameters, the optimization would possibly run for a longer time, but it should still have a fair chance at estimating the correct parameters.

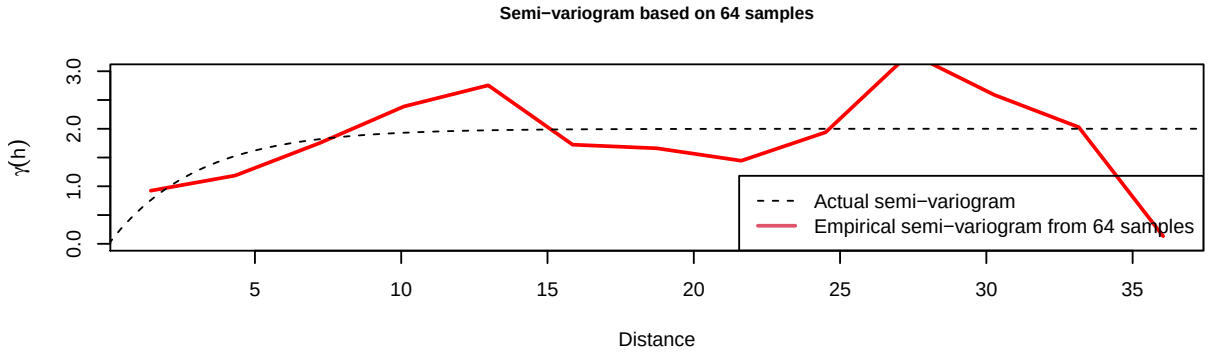
We can observe that the empirical semi-variogram we get when observing at the 36 locations becomes more spikey than the one we get when we use all locations. This is because in the latter case, we have several pairs of points with an arbitrary distance h we can use, while we risk having no locations with distance h when using only the 36 locations.

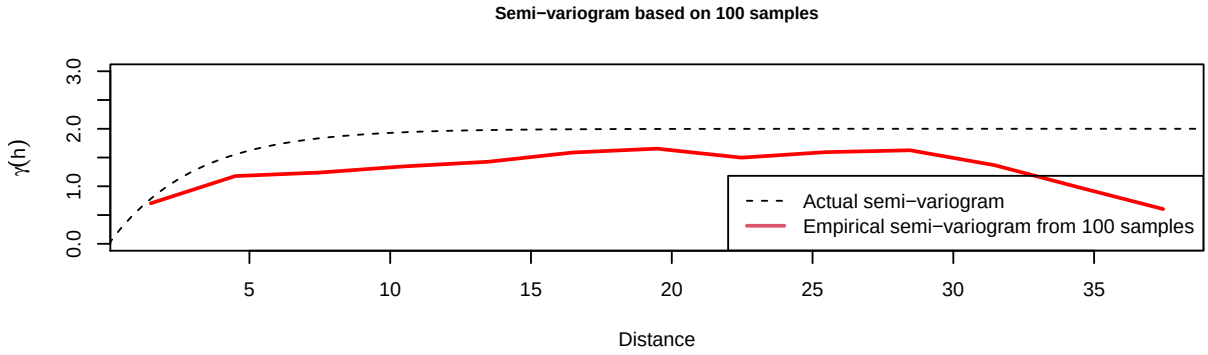
e)

We repeat the procedure with $9, 64$ and 100 locations.

```
locs_9 = sample_1_df[sample(nrow(X),size=9,replace=F),]
locs_64 = sample_1_df[sample(nrow(X),size=64,replace=F),]
locs_100 = sample_1_df[sample(nrow(X),size=100,replace=F),]



# Plot the variogram we observed

par(mfrow = c(3,1), cex.main = 0.8, cex = 0.7)

plot(variog(as.geodata(locs_9))$u,variog(as.geodata(locs_9))$v, ylim = c(0,3), type = 'l', lty = 1, lwd
lines.variomodel(sample_1, lty = 2)

legend('bottomright', c("Actual semi-variogram", "Empirical semi-variogram from 9 samples"), lty=c(2,1)


plot(variog(as.geodata(locs_64))$u,variog(as.geodata(locs_64))$v, ylim = c(0,3), type = 'l', lty = 1, l
lines.variomodel(sample_1, lty = 2)

legend('bottomright', c("Actual semi-variogram", "Empirical semi-variogram from 64 samples"), lty=c(2,1


plot(variog(as.geodata(locs_100))$u,variog(as.geodata(locs_100))$v, ylim = c(0,3), type = 'l', lty = 1,
lines.variomodel(sample_1, lty = 2)

legend('bottomright', c("Actual semi-variogram", "Empirical semi-variogram from 100 samples"), lty=c(2,
```








```
# We assume the parameters are unknown and estimate them with initial
# parameters guessed from the semi-variograms.

est_9 = likfit(as.geodata(locs_9), ini.cov.pars = c(1.7,2))

est_9_cov_parm = est_9$cov.pars
est_9_cov_mod = est_9$cov.model

est_64 = likfit(as.geodata(locs_64), ini.cov.pars = c(2,3))

est_64_cov_parm = est_64$cov.pars
est_64_cov_mod = est_64$cov.model

est_100 = likfit(as.geodata(locs_100), ini.cov.pars = c(2,2))
```

```
est_100_cov_parm = est_100$cov.pars
est_100_cov_mod = est_100$cov.model

est_covs = matrix(c(2,3,round(est_9_cov_parm[1],3), round(est_9_cov_parm[2],3), round(est_36_cov_parm[1

rownames(est_covs) = c(expression(sigma^2),'a')
colnames(est_covs) = c('True values', 'n=9', 'n=36','n=64','n=100','n = all')
```

```
est_covs
```

```
##         True values   n=9  n=36  n=64 n=100 n = all
## sigma^2           2 1.112 0.913 1.217 1.527   1.558
## a                 3 0.294 3.457 3.473 3.038   2.516
```

From the table we see that these are not terrible estimates. They are a bit wrong, compared to when we use all locations, but since we guess initial parameters that are a bit off, the parameters may converge to wrong parameters. For example, the initial conditions may have been such that we end up stuck in a local maximum (for the likelihood) yielding the wrong parameters.

The semi-variogram plots are as expected, where a lower number of locations yield a smaller set of potential distances h, and hence less information about how the variance changes over distance. In conclusion the semi-variogramsare not only more accurate, but also less fluctuating when we have observed the GRF at more locations.

f)

Throughout this problem, we have estimated empirical semi-variograms for different number of observations on a bounded domain. We have also estimated the (theoretically unknown) variance and spatial range of the underlying GRF.

When observing at varying number of locations, we noticed that fewer observed locations yield less pairs of locations of a given distance h, meaning we have less data to estimate our semi-variogram precisely. More locations yield more stable semi-variograms, giving more correct estimations of the sill, range and overall shape of the semi-variograms. Due to working on a bounded domain, we expect the highest distances to give a lot of variance, because in these extreme cases, there are always fewer pairs of locations with the given distance. As $h \to diam(\mathcal{D})$, the number of pairs goes to 0, yielding less and less data for estimating our semi-variogram.

Due to the semi-variograms being a little bit off compared to the true semi-variogram, we may encounter the problem that the likelihood based estimation of the initial parameters (σ^2, a) converge to the wrong value, as the optimization becomes stuck close to a local maximum. Hence, we may obtain faulty results from the likfit-estimation of the parameters, but since a higher number of locations observed often yield a more stable semi-variogram, we can often guess better initial conditions for our optimization problem when given more observations.